 ISSN : 1411-6286

RISC SEDERHANA DENGAN FPGA

Sofyan, S.Kom

UPT Lab Perangkat Keras, Universitas Bina Nusantara
JL. K.H. Syahdan No. 9, Jakarta, 11480, Indonesia
Email: sofyan@binus.ac.id

Abstrak

Tulisan ini ingin menyelidiki pemanfaatan teknologi FPGA untuk merancang suatu CPU dengan jenis RISC yang sederhana. CPU ini cukup sederhana sehingga dapat diimplementasikan dalam FPGA dengan kapasitas setara 10.000 gerbang logika, tetapi mencakup berbagai instruksi dasar untuk operasi matematika, logika, dan operasi percabangan serta interupsi.

Tujuan penelitian ini adalah sebagai tolok ukur kompleksitas sistem digital yang dapat diimplementasikan dalam suatu FPGA dengan kapasitas yang relatif rendah, bagi penelitian sistem digital yang lebih lanjut.

Hasil penelitian ini berupa suatu single-cycle CPU RISC 8-bit dengan hardwired control yang dapat mengerjakan 30 jenis instruksi 16-bit dengan arsitektur memory harvard. CPU ini menggunakan 69% kapasitas maksimum FPGA yang digunakan, atau setara 2.758 gerbang logika, dengan kecepatan maksimum clock yang dapat dicapai adalah sekitar 16 MHz.

Kata Kunci: cpu, fpga, risc, vhdl

1. Pendahuluan

Teknologi *Programmable Logic Device* (PLD) telah lama menjadi salah satu pilihan media untuk merancang sistem digital selain penggunaan berbagai chip digital standar dan *Application Specific Integrated Circuit* (ASIC). Hal ini dikarenakan fleksibilitas fungsi logikanya yang dapat diprogram sesuai dengan kebutuhan dan harga yang relatif murah. Beberapa jenis PLD awal yang banyak digunakan adalah *Programmable Read Only Memory* (PROM), *Programmable Array Logic* (PAL), *Generic Array Logic* (GAL), *Programmable Logic Array* (PLA), dan sebagainya. Chip-chip PLD awal ini umumnya memiliki kapasitas sekitar beberapa ratus gerbang logika.

Seiring dengan berkembangnya teknologi dan meningkatnya kompleksitas dari sistem digital, maka fleksibilitas dan kapasitas dari berbagai PLD juga ikut meningkat sesuai kebutuhan. Dimulai dengan munculnya *Complex Programmable Logic Device* (CPLD) dengan kapasitas berkisar beberapa ribu hingga beberapa puluh ribu gerbang logika, dilanjutkan dengan munculnya *Field Programmable Gate Array* (FPGA) dengan kapasitas berkisar beberapa ribu hingga beberapa juta gerbang logika.

FPGA merupakan PLD yang dibangun dari kumpulan sel fungsi logika dasar yang dapat diprogram. Sel-sel logika ini terhubung satu sama lain melalui suatu jaringan interkoneksi yang juga dapat diprogram. Istilah "*Field Programmable*" mengandung arti bahwa chip ini dapat diprogram di luar dari pabriknya, atau dengan kata lain, dapat diprogram oleh pengguna. Berbeda dengan ASIC yang keseluruhan proses produksi harus dilakukan di pabrik, atau beberapa generasi *gate array* sebelumnya yang harus melalui proses akhir di pabrik untuk mendapatkan fungsi yang diinginkan.

Karena sifatnya yang programmable, maka FPGA, seperti halnya PLD yang lain, dapat diproduksi dalam jumlah banyak untuk aplikasi yang luas, sehingga harganya menjadi relatif murah. Akan tetapi berbeda dengan generasi PLD sebelumnya, FPGA memiliki kapasitas yang jauh lebih besar, sehingga dapat digunakan untuk menggantikan fungsi kompleks yang sebelumnya hanya didominasi oleh ASIC yang mahal.

Apabila dibandingkan antara FPGA dan ASIC, maka masing-masing memiliki kelebihan dan kekurangan. Suatu FPGA dengan fungsi yang sama dengan ASIC, umumnya memiliki kecepatan yang lebih rendah dan membutuhkan daya listrik yang lebih besar. Akan tetapi waktu yang dibutuhkan

untuk perancangan dan produksi sistem dengan FPGA jauh lebih cepat dibandingkan dengan ASIC, selain itu biaya perancangan dan produksi sistem dengan FPGA pada umumnya lebih murah daripada ASIC.

Dengan karakteristik tersebut, FPGA sangat sesuai digunakan sebagai media perancangan sistem digital untuk penelitian. Namun yang sering menjadi pertanyaan adalah seberapa kompleks sistem digital yang dapat diimplementasikan dalam suatu FPGA, karena umumnya kita tidak memiliki referensi hubungan kompeksitas sistem dengan jumlah gerbang atau sel logika yang dibutuhkan. Oleh sebab itu penelitian ini dilakukan untuk memberikan contoh sistem digital yang yang dapat diimplementasikan dalam suatu FPGA.

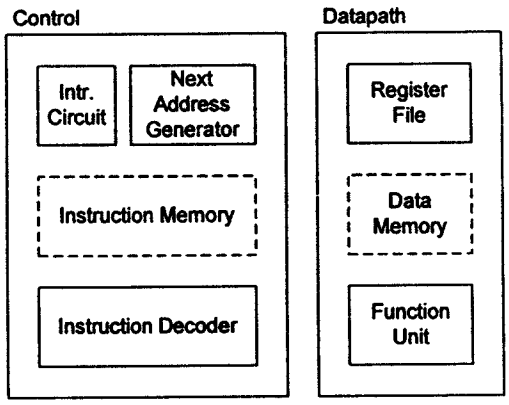
Central Processing Unit (CPU) merupakan suatu sistem digital yang dapat dijumpai pada hampir semua perangkat digital yang kita kenal di sekitar kita. Pada perangkat-perangkat tersebut, CPU berfungsi sebagai "otak" dari semua fungsi yang dapat dikerjakannya. Karena sifatnya yang umum, maka pada penelitian ini, CPU dijadikan sebagai tolok ukur mengenai kompleksitas suatu sistem digital.

Pada penelitian ini dirancang suatu CPU sederhana yang berbasiskan *Reduced Instruction Set Computer* (RISC) dengan sifat *single-cycle*. Arsitektur RISC dipilih karena kesederhanaanya dapat mewakili bentuk dasar dari sistem digital sinkron. Istilah *single-cycle* di sini berarti CPU yang dimaksud menyelesaikan setiap instruksi dalam satu cycle sinyal clock. Lebih lengkapnya, spesifikasi dari CPU yang ingin dirancang adalah sebagai berikut:

- CPU RISC 8-bit
- Arsitektur memory *harvard*
- *Single-cycle* CPU
- Memiliki instruksi dasar untuk logika dan aritmatika
- Dapat menerima sinyal interupsi eksternal
- Lebar instruksi 16-bit

Chip FPGA yang digunakan pada penelitian ini adalah Xilinx Spartan XCS10PC84, yang memiliki 196 sel logika yang disebut *Configurable Logic Block* (CLB) atau setara dengan 10.000 gerbang logika. Chip ini memiliki total 616 *Flip-Flop* (FF) dengan jumlah pin I/O yang dapat digunakan sebanyak 61 pin. Kecepatan maksimum yang dapat dicapai adalah sekitar 80MHz.

Blok diagram dari CPU yang dirancang digambarkan pada Gambar 1, secara umum terdiri dari dua bagian, yaitu Datapath dan Control yang akan dijelaskan pada bagian berikutnya.

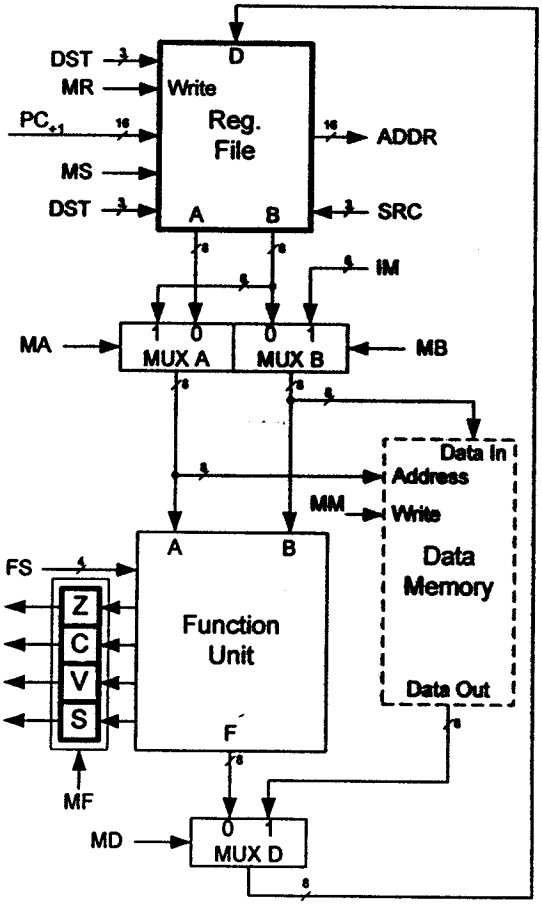


Gambar 1. Blok Diagram CPU

2. Datapath

Bagian Datapath yang terdiri dari Register File dan Function Unit yang berhubungan dengan *Data Memory* eksternal sebagai sumber data atau tempat penyimpanan hasil operasi. Semua aliran data dan operasi terhadap data dilakukan di dalam datapath. Selain data memory, sumber atau tujuan data yang diproses dapat pula berupa input atau output dari komponen I/O eksternal yang dihubungkan

seperti data memory, tetapi dengan pembagian alamat antara memori dan masing-masing I/O. Blok diagram yang lebih detil dari bagian datapath ditunjukkan pada Gambar 2. Pada gambar tersebut, semua komponen dengan garis kotak yang tebal memiliki register di dalamnya, selebihnya adalah rangkaian kombinasional.



Gambar 2. Blok Diagram Datapath

Pada datapath, register file digunakan untuk menyimpan data input atau hasil operasi sementara, dan menampung alamat data pada data memory. Selain itu register file juga digunakan untuk menampung alamat instruksi untuk operasi pemanggilan prosedur.

Sedangkan function unit digunakan untuk melakukan operasi logika, pergeseran biner, dan aritmatika. Status hasil operasi aritmatika disimpan pada Status Flag Z, C, V, dan S yang kemudian akan digunakan sebagai kondisi untuk melakukan operasi percabangan.

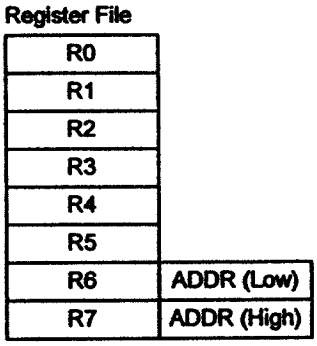
Dua multiplexer 2 to 1 (Mux A dan Mux B) digunakan untuk menentukan aliran data 8-bit yang menjadi input bagi function unit dan data memory. Sedangkan multiplexer MUX D digunakan untuk memilih hasil operasi dari function unit atau pembacaan dari data memory (atau I/O bila ada) yang akan disimpan pada register file.

Secara keseluruhan operasi dari datapath dikendalikan oleh output dari bagian control, dalam bentuk sinyal kontrol, antara lain: IM, FS, DST, SRC, MA, MB, MD, MF, MM, MR, dan MS. Terkecuali PC_{+1} dan ADDR, berfungsi untuk menyimpan dan mengembalikan alamat instruksi.

2.1. Register File

Seperti ditunjukkan oleh Gambar 3, di dalam register file terdapat delapan buah register 8-bit yang diberi nama R0 hingga R7. Khusus untuk R6 dan R7 digunakan juga untuk menampung *return*

address dari PC_{+1} apabila ada operasi pemanggilan prosedur, ditandai dengan aktifnya sinyal kontrol MS.



Gambar 3. Daftar Register pada Register File

Dalam satu saat terdapat dua register yang dapat dibaca sekaligus untuk dioperasikan, dengan dikendalikan oleh sinyal kontrol DST dan SRC. Kedua sinyal kontrol tersebut digunakan untuk memilih dua dari delapan register untuk dikeluarkan pada output A dan B register file. Sinyal kontrol DST juga sekaligus digunakan untuk menentukan register yang akan digunakan untuk menyimpan hasil operasi pada input D register file, apabila ada operasi yang mengembalikan hasilnya ke register, ditandai dengan aktifnya sinyal kontrol MR.

2.2. Function Unit

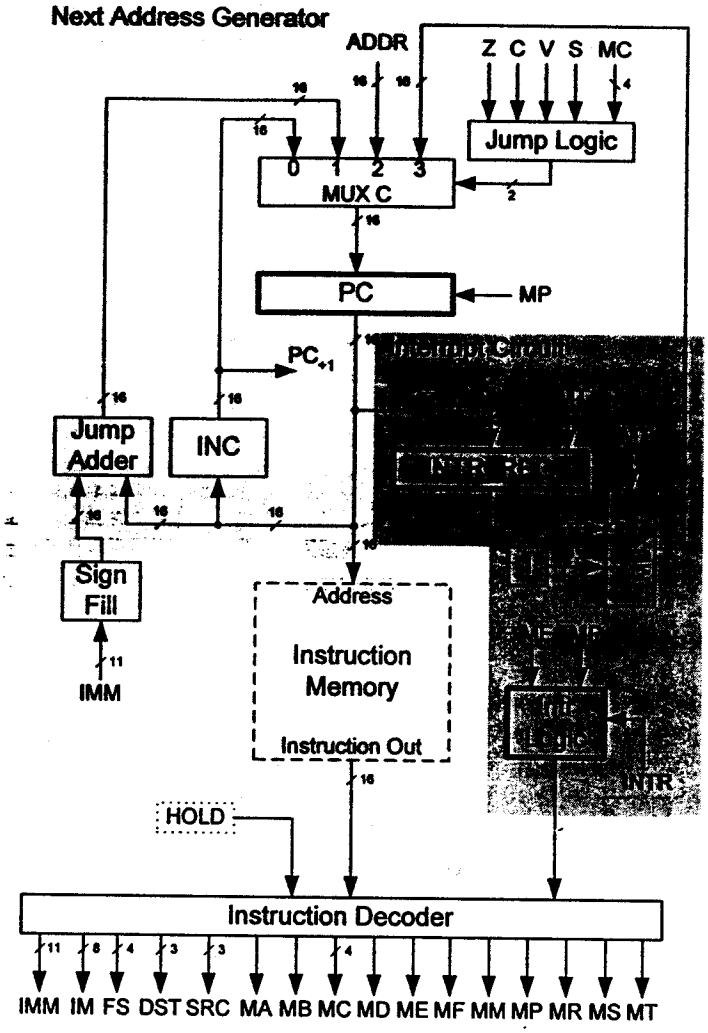
Function unit melakukan operasi logika, pergeseran biner atau aritmatika terhadap input A dan B yang diterimanya. Jenis operasi yang dikerjakan dikendalikan oleh sinyal kontrol FS. Secara keseluruhan terdapat 11 jenis operasi yang dapat dikerjakan oleh function unit, antara lain:

- F ← A + B
- F ← A – B
- F ← A + 1
- F ← A – 1
- F ← A AND B
- F ← A OR B
- F ← A XOR B
- F ← NOT A
- F ← B
- F ← Rotate Right A
- F ← Rotate Left A

3. Control

Bagian control berfungsi untuk mengendalikan seluruh operasi dari CPU dengan membaca instruksi dari programmer, menterjemahkannya, dan menghasilkan sinyal kontrol yang akan mengendalikan datapath untuk mengerjakan operasi sesuai dengan instruksi yang diterima. Control unit juga mengendalikan alamat instruksi yang akan diambil pada operasi percabangan, pemanggilan prosedur, dan interupsi.

Control unit terutama terdiri dari *Instruction Decoder*, *Next Address Generator*, dan *Interrupt Circuit*. Termasuk di dalam control unit adalah kumpulan instruksi dari programmer yang disimpan dalam *Instruction Memory* eksternal yang umumnya berupa chip ROM. Blok diagram lebih detil dari control diberikan pada Gambar 4.

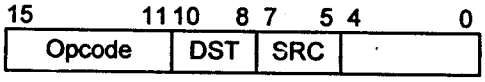


Gambar 4. Blok Diagram Control

Pada Gambar 4 di atas, komponen-komponen dengan garis tebal memiliki register di dalamnya, selebihnya adalah rangkaian kombinasional. Semua komponen pada daerah yang berwarna abu-abu muda adalah bagian dari next address generator, sedangkan semua komponen pada daerah yang berwarna abu-abu tua adalah bagian dari interrupt circuit.

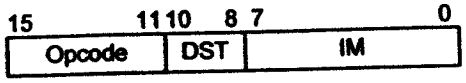
3.1. Instruction Set

Instruksi yang dapat dikerjakan oleh RISC pada umumnya terbatas, dengan mode pengalamatan yang juga terbatas jumlahnya. Pada penelitian ini format instruksi dibatasi maksimum memiliki dua operand, yang disebut DST dan SRC. Mode pengalamatan yang dapat dilakukan hanya *Register*, *Register Indirect*, *Immediate*, dan *Relative*.

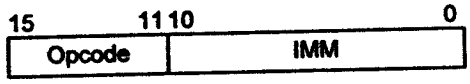


(a) Format Instruksi untuk Operasi Register ke Register

 ISSN : 1411-6286



(b) Format Instruksi untuk Operasi dengan Short Immediate



(c) Format Instruksi untuk Operasi dengan Long Immediate

Gambar 5. Format Instruksi

Terdapat 3 jenis format instruksi yang digunakan seperti terlihat pada Gambar 5. Format pada Gambar 5(a) adalah untuk operasi dengan dua operand register, dapat melakukan mode pengalamatan register dan register indirect. Format pada Gambar 5(b) digunakan untuk operasi dengan operand register dan immediate 8-bit (*Short Immediate*), dapat melakukan mode pengalamatan immediate. Format pada Gambar 5(c) digunakan untuk operasi dengan operand immediate 11-bit (*Long Immediate*) yang digunakan sebagai alamat, dapat melakukan mode pengalamatan relative. Yang membedakan jenis operasi dan format yang digunakan untuk masing-masing operasi adalah *Opcode* dengan lebar 5-bit.

3.2. Instruction Decoder

Instruction decoder merupakan komponen penting yang berfungsi untuk menterjemahkan instruksi dari instruction memory menjadi sinyal kontrol yang akan mengendalikan operasi yang dikerjakan oleh CPU. Input *Hold* pada instruction decoder merupakan fungsi tambahan untuk menghentikan seluruh operasi CPU. Sedangkan input dari *Intr Logic* pada instruction decoder akan menyebabkan aktifnya sinyal kontrol untuk melakukan percabangan ke sub rutin interupsi.

Secara keseluruhan terdapat 16 sinyal kontrol yang dihasilkan oleh instruction decoder, dimana 11 diantaranya digunakan untuk mengendalikan datapath, dan sisanya untuk mengendalikan bagian control sendiri. Fungsi masing-masing sinyal kontrol tersebut dijelaskan pada Tabel 1.

Tabel 1. Sinyal kontrol dan Fungsinya

Sinyal	Fungsi
IMM	Alamat 11-bit untuk operasi percabangan
IM	Data immediate 8-bit
FS	Menentukan jenis operasi pada Function Unit
DST	Alamat register target sekaligus operand A
SRC	Alamat register untuk operand B
MA	Memilih operand A dari output A atau B dari Register File
MB	Memilih operand B dari output B Register File atau IM
MC	Menentukan loading terhadap PC
MD	Memilih output Function Unit atau Data Memory
ME	Menyatakan End of Interrupt
MF	Memperbolehkan penulisan ke Status Flag
MM	Memperbolehkan penulisan ke Data Memory
MP	Memperbolehkan penulisan ke PC
MR	Memperbolehkan penulisan ke Register File
MS	Menandakan terjadinya pemanggilan prosedur
MT	Menandakan terjadinya interupsi

3.3. Next Address Generator

Next address generator berfungsi untuk menentukan alamat instruksi berikutnya yang harus dibaca dari instruction memory, terutama untuk melakukan instruksi untuk percabangan maupun dari interupsi eksternal. Komponen-komponen dalam next address generator adalah: *Program Counter* (PC), *Jump Adder*, INC, MUX C, *Jump Logic*, dan *Sign Fill*.

PC adalah suatu counter yang menyimpan alamat instruksi yang sedang dikerjakan. Kemungkinan alamat yang berikutnya akan dikerjakan adalah:

- Alamat yang sekarang ditambah satu Á output INC
- Alamat yang sekarang ditambah IMM (telah melalui Sign Fill) Á output Jump Adder
- Alamat return address dari sub rutin prosedur Á output ADDR dari register file
- Alamat 0001 heksadesimal yang merupakan vektor interupsi Á output interrupt circuit

Dari keempat kemungkinan itu akan dipilih salah satu oleh MUX C, berdasarkan output dari jump logic. Jump logic sendiri bekerja berdasarkan sinyal kontrol MC yang menentukan apakah: MUX C akan memilih output dari INC atau output dari Jump Adder atau ADDR atau output dari interrupt logic. Khusus untuk instruksi percabangan dengan kondisi, sinyal kontrol MC akan menentukan apakah memilih output dari jump adder atau output dari INC bergantung pada salah satu kondisi: Z, C, V, atau S.

3.4. Interrupt Circuit

Interrupt circuit berfungsi untuk mendeteksi adanya sinyal interupsi dan mengendalikan proses percabangan ketika terjadi interupsi. Deteksi adanya sinyal interupsi dilakukan oleh *Intr Logic* yang mendeteksi terjadinya perubahan sinyal pada pin INTR dari *low* ke *high*. Ketika terjadi terjadi permintaan interupsi, instruction decoder akan mengaktifkan sinyal MT, yang memperbolehkan register INTR_REG untuk menyimpan alamat instruksi yang harus dikerjakan ketika selesai mengerjakan sub rutin interupsi, sekaligus mengirimkan nilai 0001 heksadesimal ke MUX C untuk melompat ke vektor interupsi.

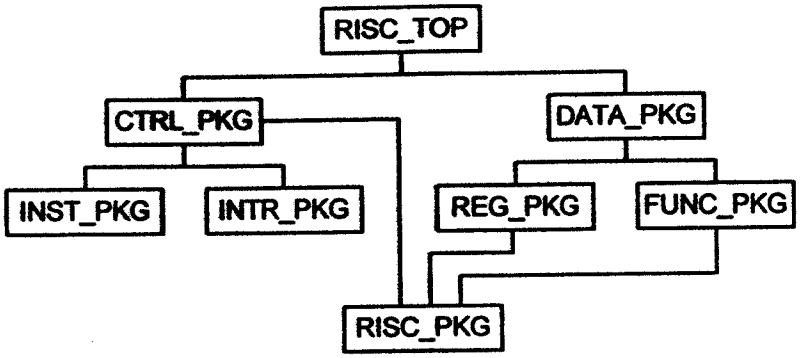
Sub rutin interupsi akan diakhiri dengan instruksi *Interrupt Return* yang akan mengaktifkan sinyal kontrol ME yang berarti akhir dari pengerjaan sub rutin interupsi. Instruksi ini akan menghasilkan sinyal kontrol MC yang akan mengendalikan jump logic agar MUX C memilih output dari INTR_REG, untuk melanjutkan pengerjaan program yang disela oleh terjadinya interupsi sebelumnya.

Karena untuk menyimpan alamat kembali dari sub rutin interupsi hanya ada sebuah register, yaitu INTR_REG, maka tidak dapat terjadi interupsi yang lain sebelum interupsi yang sebelumnya selesai. Dengan kata lain tidak dapat terjadi *nested interrupt*.

4. Implementasi

Perancangan sistem dilakukan menggunakan *VHSIC Hardware Description Language* (VHDL). Proses perancangan, sintesis, simulasi fungsional, dan implementasi dikerjakan pada software Xilinx Foundation Series 2.1i Student Edition.

Hirarki rancangan dalam VHDL ditunjukkan pada Gambar 6, dimana RISC_TOP merupakan *top level file* dari desain. Kemudian desain dijelaskan oleh dua bagian, yaitu CTRL_PKG untuk mendeskripsikan bagian control, dan DATA_PKG untuk mendeskripsikan bagian datapath. CTRL_PKG dijelaskan oleh INST_PKG untuk mendeskripsikan instruction decoder, dan INTR_PKG untuk mendeskripsikan interrupt circuit. Sedangkan bagian DATA_PKG dijelaskan oleh REG_PKG untuk mendeskripsikan register file, dan FUNC_PKG untuk mendeskripsikan function unit. Beberapa fungsi digital dasar dikumpulkan dalam RISC_PKG yang digunakan pada CTRL_PKG, REG_PKG, dan FUNC_PKG.



Gambar 6. Hirarki Perancangan dalam VHDL

Total instruksi yang dapat dikerjakan oleh CPU RISC sederhana ini adalah sebanyak 30 instruksi, yang terdiri dari 20 instruksi transfer dan manipulasi data termasuk operasi dengan operand immediate, kemudian terdapat 9 instruksi percabangan termasuk penanganan prosedur dan interupsi, dan terakhir adalah sebuah instruksi *no operation*. Detil instruksi yang dapat dikerjakan ditunjukkan pada Tabel 2.

Table 2. Daftar Instruksi CPU RISC Sederhana

Operasi	Symbol	Register Transfer
Add	ADD	R[DST] ← R[DST] + R[SRC]
Subtract	SUB	R[DST] ← R[DST] – R[SRC]
Increase	INC	R[DST] ← R[DST] + 1
Decrease	DEC	R[DST] ← R[DST] – 1
AND	AND	R[DST] ← R[DST] ^ R[SRC]
OR	OR	R[DST] ← R[DST] v R[SRC]
XOR	XOR	R[DST] ← R[DST] ⊕ R[SRC]
Complement	NOT	R[DST] ← $\overline{R[DST]}$
Interrupt Return	IRT	PC ← INTR_REG, INTR_REG ← 0001
Compare	CMP	Z,C,V,S ← status R[DST] – R[SRC]
Move	MOV	R[DST] ← R[SRC]
Store	ST	M[R[DST]] ← R[SRC]
Load	LD	R[DST] ← M[R[SRC]]
Rotate Right	RR	R[DST] ← rr R[DST]
Rotate Left	RL	R[DST] ← rl R[DST]
Jump Immediate	JMP	PC ← PC + sf IMM
Jump if Equal	JE	Z : PC ← PC + sf IMM, PC ← PC_{+1}
Jump if Above	JA	$\overline{C+Z}$: PC ← PC + sf IMM, PC ← PC_{+1}
Jump if Below	JB	C : PC ← PC + sf IMM, PC ← PC_{+1}
Jump if Greater	JG	$\overline{(S \oplus V)+Z}$: PC ← PC + sf IMM, PC ← PC_{+1}
Jump if Less	JL	S⊕V : PC ← PC + sf IMM, PC ← PC_{+1}
Jump Register	JMR	PC ← R[ADDR]
Call Procedure	CAL	R[ADDR] ← PC_{+1}, PC ← PC + sf IMM
Add Immediate	ADI	R[DST] ← R[DST] + IM
Subtract Immediate	SBI	R[DST] ← R[DST] – IM
Move Immediate	MVI	R[DST] ← IM
AND Immediate	ANI	R[DST] ← R[DST] ^ IM
OR Immediate	ORI	R[DST] ← R[DST] v IM
XOR Immediate	XOI	R[DST] ← R[DST] ⊕ IM
No Operation	NOP	None

Implementasi dengan target FPGA Xilinx Spartan XCS10PC84, memerlukan 137 CLB dari 196 CLB, atau 69% kapasitas FPGA tersebut. Jumlah FF yang digunakan untuk register adalah sebanyak 102. Rangkaian kombinasional pada rancangan menggunakan 216 buah *Look Up Table* (LUT) 4 input, dan 70 buah LUT 3 input. Jumlah pin I/O yang digunakan adalah 60 dari total 61 pin yang tersedia, atau 98% jumlah pin yang dimiliki. Selain itu rancangan ini juga menggunakan 128 dari 224 *tri-state buffer*, terutama untuk mengimplementasikan multiplexer. Keseluruhan *resource* yang digunakan pada rancangan ini setara dengan 2.758 gerbang logika.

Dari sisi performa, analisa timing dari rancangan yang telah diimplementasikan dalam FPGA menghasilkan periode delay maksimum dari dari satu cycle sinyal clock ke cycle yang lain sebesar 62,264ns, sehingga kecepatan maksimum clock yang dapat digunakan adalah 16,061MHz, atau 16,061 juta instruksi per detik, karena satu cycle CPU ini sama dengan satu periode sinyal clock.

5. Kesimpulan

Dari hasil implementasi, terlihat bahwa CPU RISC sederhana yang dirancang ini walaupun hanya memiliki instruksi-instruksi dasar, namun bisa dikatakan telah cukup untuk mengerjakan berbagai fungsi logika maupun aritmatika umum. Kecepatan yang dapat dicapai setara atau bahkan di atas kebanyakan mikrokontroler standar. Dan keseluruhan CPU RISC sederhana ini menggunakan 69% dari kapasitas maksimum dari suatu FPGA dengan kapasitas setara 10 ribu gerbang logika.

Chip FPGA yang digunakan pada rancangan ini termasuk dalam kategori FPGA dengan kapasitas dan performa yang rendah. Sehingga dapat disimpulkan bahwa fungsi yang jauh lebih kompleks dapat diimplementasikan pada FPGA dengan kapasitas ratusan ribu hingga jutaan gerbang logika dengan kecepatan mencapai ratusan MHz. Performa yang lebih tinggi dapat dicapai apabila proses dapat dioptimasi dengan desain arsitektur yang paralel.

6. Daftar Pustaka

[1] Mano, M. Morris and Kime, Charles R., *Logic and Computer Design Fundamentals*, Prentice-Hall International, Inc.

[2] Anonim, *Spartan and Spartan-XL Families Field Programmable Gate Arrays*, Xilinx, Inc., 2000.

[3] Yalamanchili, Sudhakar, *Introductory VHDL From Simulation to Synthesis*. Prentice-Hall, Inc., 2001.

[4] Anonim, *Foundation Series 2.1i Software Documentation*. Xilinx, Inc., 2000.